PALM INTRANET

Day : Saturday
Date: 7/7/2007
Time: 17:16:51

Continuity Information for 10/751236

Parent Data
10751236
is a continuation of 10001095

Child Data
No Child Data

Appln Info | Contents | Petition Info | Atty/Agent Info | **Continuity/Reexam** | Foreign Data | Inve

Search Another: Application # Search **or Patent#** Search
PCT / / Search **or PG PUBS #** Search
Attorney Docket # Search
Bar Code # Search

To go back use Back button on your browser toolbar.

Back to PALM | ASSIGNMENT | OASIS | Home page

EAST Search History

Ref #	Hits	Search Query	DBs	Default Operator	Plurals	Time Stamp
L1	135	(generat$4 same set$1 same (quer$4 sql$1) same table$1 same hierarch$5)	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2007/07/07 15:57
L2	3800	(parent$1 same child$1 same table$1)	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2007/07/07 15:58
L3	1388	(parent$1 same child$1 same table$1) and (hierarch$5 same table)	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2007/07/07 15:59
L4	27	(parent$1 same child$1 same table$1) and (hierarch$5 same table same integrat$5)	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2007/07/07 15:59
L5	44738	"707"/$.ccls.	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2007/07/07 16:00
L6	17	4 and 5	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2007/07/07 16:01
L7	101	generat$4 same ("select statement" with clause)	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2007/07/07 16:03
L8	74	(generat$4 same ("select statement" with clause)) and record$1	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2007/07/07 16:03
L9	69	(generat$4 same ("select statement" with clause)) and record$1 and (number$1 or "maximum number")	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2007/07/07 16:04

EAST Search History

L10	16	(generat$4 same ("select statement" with clause)) and record$1 and (number$1 or "maximum number") and ("insert operation")	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2007/07/07 16:05
L11	0	(generat$4 same ("select statement" with clause)) and record$1 and (number$1 or "maximum number") and ("insert operation") and (set with instance$1)	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2007/07/07 16:05
L12	16	(generat$4 same ("select statement" with clause)) and record$1 and (number$1 or "maximum number") and ("insert operation") and set$1	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2007/07/07 16:06
L13	15	(generat$4 same ("select statement" with clause)) and record$1 and (number$1 or "maximum number") and ("insert operation") and set$1 and instance$1	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2007/07/07 16:06

IEEE Xplore® RELEASE 2.3

Home | Login | Logout | Access Information | Alerts | Sitema

Welcome United States Patent and Trademark Office

□ Search Results · BROWSE SEARCH IEEE XPLORE GUIDE SUPPORT

Results for "((((generat* and select and statement)<in>metadata))<and>((generat* and select and state..."
Your search matched **2** of **22** documents.
A maximum of **100** results are displayed, **25** to a page, sorted by **Relevance** in **Descending** order.

e-mail printe

» Search Options

View Session History

New Search

Modify Search

((((generat* and select and statement)<in>metadata))<and>((generat* and select and Search

□ Check to search only within this results set

Display Format: ● Citation ○ Citation & Abstract

» Key

IEEE JNL	IEEE Journal or Magazine
IET JNL	IET Journal or Magazine
IEEE CNF	IEEE Conference Proceeding
IET CNF	IET Conference Proceeding
IEEE STD	IEEE Standard

view selected items Select All Deselect All

□ 1. **Generating Relational Database Transactions From Recursive Functions Defined on EB^3 Traces**
Gervais, F.; Frappier, M.; Laleau, R.;
Software Engineering and Formal Methods, 2005. SEFM 2005. Third IEEE International Conference on
7-9 Sept. 2005 Page(s):117 - 126
Digital Object Identifier 10.1109/SEFM.2005.23

AbstractPlus | Full Text: PDF(328 KB) IEEE CNF
Rights and Permissions

□ 2. **Specifying Ada server tasks with executable formal grammars**
Hemmendinger, D.;
Software Engineering, IEEE Transactions on
Volume 16, Issue 7, July 1990 Page(s):741 - 754
Digital Object Identifier 10.1109/32.56100

AbstractPlus | Full Text: PDF(1184 KB) IEEE JNL
Rights and Permissions

Help Contact Us Privacy & Security

Indexed by Inspec®

< Back to Previc

Specifying Ada server tasks with executable formal grammars

Hemmendinger, D.
Dept. of Comput. Sci., Union Coll., Schenectady, NY;

This paper appears in: **Software Engineering, IEEE Transactions on**
Publication Date: Jul 1990
Volume: 16, Issue: 7
On page(s): 741-754
ISSN: 0098-5589
References Cited: 27
CODEN: IESEDJ
INSPEC Accession Number: 3724562
DOI: 10.1109/32.56100
Posted online: 2002-08-06 17:23:49.0

Abstract

The author shows how a class of concurrent programming problems can be specified with formal grammars. These grammars, more powerful than path expressions, translate readily into Ada server tasks using the rendezvous and select-statement, though they may also be applied to other synchronization constructs. The grammars may be used to clarify informal specifications, to compare different specifications, and to analyze the behavior of implementations of such specifications. They may also be easily converted into Prolog programs that can be executed to generate the strings of events accepted by a grammar or by the Ada task being modeled. The automated translation from Ada to such grammars, and from grammatical specifications to Ada is discussed. The former facilitates the analysis of Ada programs; the latter yields Ada code of high quality

Index Terms
Inspec

Controlled Indexing
Ada formal specification grammars parallel programming

Non-controlled Indexing
Ada server tasks specification Prolog programs concurrent programming problems executable formal grammars synchronization constructs

Author Keywords
Not Available

References

No references available on IEEE Xplore.

Citing Documents

No citing documents available on IEEExplore.

Indexed by
Inspec®

IEEE Xplore® RELEASE 2.3

Hoı

□ AbstractPlus - Print Format

< Back to Previc

Generating Relational Database Transactions From Recursive Functions Defined on EB^3 Traces

Gervais, F. Frappier, M. Laleau, R.
Universite de Sherbrooke, Canada;

This paper appears in: **Software Engineering and Formal Methods, 2005. SEFM 2005. Third IEEE International Conference on**
Publication Date: 7-9 Sept. 2005
On page(s): 117- 126
ISBN: 0-7695-2435-4
DOI: 10.1109/SEFM.2005.23
Posted online: 2006-01-16 09:05:19.0

Abstract

EB^3 is a trace-based formal language created for the specification of information systems (IS). Attributes, linked to entities and associations of an IS, are computed in EB^3 by recursive functions on the valid traces of the system. We aim at synthesizing relational database transactions that correspond to EB^3 attribute definitions. Each EB3 action is translated into a transaction. EB^3 attribute definitions are analysed to determine the key values affected by each action. Some key values are retrieved from SELECT statements that correspond to first-order predicates in EB^3 attribute definitions. To avoid problems with the sequencing of SQL statements in the transactions, temporary variables and/or tables are introduced for these key values. Generation of DELETE statements is straightforward, but distinguishing updates from insertions of tuples requires more analysis.

Index Terms
Inspec

Controlled Indexing
Not Available

Non-controlled Indexing
Not Available

Author Keywords
Not Available

References

No references available on IEEE Xplore.

Citing Documents

No citing documents available on IEEEXplore.

Web Images Video News Maps Gmail more ▾ Sign in

Google

selecting set of instance data set record table | Search | Advanced Search
Preferences

Web Results **1** - **10** of about **1,710,000** for **selecting set of instance data set record table generating select statemer**

Tip: Save time by hitting the return key instead of clicking on "search"

Proposion - Visual Studio.NET Integration in Proposion N2N 2.0
Finally, you have the option of **selecting data records** by doing a direct .... This tool lets you **select** the Typed **Data Set** (if any) you want to use and ...
www.proposion.com/.../0e406e203b53eda385256b6c0045401c/
611870cf474758be8525700f00355961!OpenDocument - 44k - Cached - Similar pages

OpenBase Tips Welcome to OpenBase! Here are some helpful tips to ...
Which is faster: using views or **tables**? **Selecting data** through a view ... A varchar, or char type, can be used to collect a **set** of related **record** keys for ...
openbase.com/help/KnowledgeBase/100_Introduction/102_OpenBaseTips.html - 39k -
Cached - Similar pages

Rudimentary SQL Tuning for ODS Users
SELECT STATEMENT Cost = 1 (1) SORT(UNIQUE) 2 (1) FILTER 3 (1) NESTED ...
records via index (don't throw out unmatched) 9 for every **record** in resulting **set**, ...
www.alaska.edu/oit/training/Documentation_docs/tuning.html - 14k -
Cached - Similar pages

Binding Controls with VB .NET > Design Time Binding
If the **table** used in the **SELECT statement** does not have a primary key, ... Open the **Data** menu, and **select Generate Dataset** to see the dialog box shown in ...
www.quepublishing.com/articles/article.asp?p=26956 - 31k - Cached - Similar pages

Automatically **Generating** Commands
To automatically **generate** SQL statements for a DataAdapter, first **set** the ... The following example recreates the Customers **table** in the custDS **dataset**. ...
msdn2.microsoft.com/en-us/library/tf579hcz(VS.80).aspx - 27k - Cached - Similar pages

> Automatically Generated Commands
> To automatically **generate** SQL statements for a DataAdapter, first **set** the SelectCommand ... **Generate** an SQL **Statement**") and recreates the Customers **table**, ...
> msdn2.microsoft.com/en-us/library/tf579hcz(VS.71).aspx - 23k - Cached - Similar pages

MySQL AB :: MySQL 5.0 Reference Manual :: 23.2.3.2 Using ...
Data.DataSet , a connection string, a query string that is a SQL **SELECT statement**, and a string that is the name of the database **table**. ...
mysql.org/doc/refman/5.0/en/connector-net-examples-mysqlcommandbuilder.html - 35k -
Cached - Similar pages

LinuxCBT DBMS Edition, Linux Training, Computer Based Training ...
Discuss common DML statements; Focus on **SELECT** DML **statement** ... Compare & contrast **table** & index storage requirements; **Generate data-set** to be imported ...
www.linuxcbt.com/products_linuxcbt_dbms_edition.php - 52k - Cached - Similar pages

DB2eDataAdapter members
Table 3 describes public **instance** methods provided by DB2eDataAdapter. ... Gets the parameters **set** by the user when executing an SQL **SELECT statement**. ...
publib.boulder.ibm.com/infocenter/db2e82/topic/com.ibm.db2e.doc/dbeapr1804.html - 11k -
Cached - Similar pages

> DB2eDataAdapter Members

Fill (inherited from DbDataAdapter), Adds or refreshes rows to a **DataSet** or ... Gets the parameters **set** by the user when executing an SQL **SELECT statement**. ...
publib.boulder.ibm.com/infocenter/iwedhelp/v6r0/topic/com.ibm.db2e.doc/dbeapr1804.html - 9k - Cached - Similar pages
[More results from publib.boulder.ibm.com]

1 2 3 4 5 6 7 8 9 10 **Next**

selecting set of instance data set rec Search

Search within results | Language Tools | Search Tips | Dissatisfied? Help us improve

.NET Framework Developer's Guide

Automatically Generating Commands

When the **SelectCommand** property is dynamically specified at run time, such as through a query tool that takes a textual command from the user, you may not be able to specify the appropriate **InsertCommand**, **UpdateCommand**, or **DeleteCommand** at design time. If your DataTable [http://msdn2.microsoft.com/en-us/library/system.data.datatable(VS.80).aspx] maps to or is generated from a single database table, you can take advantage of the DbCommandBuilder [http://msdn2.microsoft.com/en-us/library/system.data.common.dbcommandbuilder(VS.80).aspx] object to automatically generate the **DeleteCommand**, **InsertCommand**, and **UpdateCommand** of the DbDataAdapter [http://msdn2.microsoft.com/en-us/library/system.data.common.dbdataadapter(VS.80).aspx] .

As a minimum requirement, you must set the **SelectCommand** property in order for automatic command generation to work. The table schema retrieved by the **SelectCommand** property determines the syntax of the automatically generated INSERT, UPDATE, and DELETE statements.

The **DbCommandBuilder** must execute the **SelectCommand** in order to return the metadata necessary to construct the INSERT, UPDATE, and DELETE SQL commands. As a result, an extra trip to the data source is necessary, and this can hinder performance. To achieve optimal performance, specify your commands explicitly rather than using the **DbCommandBuilder**.

The **SelectCommand** must also return at least one primary key or unique column. If none are present, an **InvalidOperation** exception is generated, and the commands are not generated.

When associated with a **DataAdapter**, the **DbCommandBuilder** automatically generates the **InsertCommand**, **UpdateCommand**, and **DeleteCommand** properties of the **DataAdapter** if they are null references. If a **Command** already exists for a property, the existing **Command** is used.

Database views that are created by joining two or more tables together are not considered a single database table. In this instance you cannot use the **DbCommandBuilder** to automatically generate commands; you must specify your commands explicitly. For information about explicitly setting commands to resolve updates to a **DataSet** back to the data source, see Updating Data Sources with DataAdapters [http://msdn2.microsoft.com/en-us/library/33y2221y (VS.80).aspx] .

You might want to map output parameters back to the updated row of a **DataSet**. One common task would be retrieving the value of an automatically generated identity field or time stamp from the data source. The **DbCommandBuilder** will not map output parameters to columns in an updated row by default. In this instance you must specify your command explicitly. For an example of mapping an automatically generated identity field back to a column of an inserted row, see

Retrieving Identity or Autonumber Values [http://msdn2.microsoft.com/en-us/library/ks9f57t0(VS.80).aspx] .

Rules for Automatically Generated Commands

The following table shows the rules for how automatically generated commands are generated.

Command	Rule
InsertCommand	Inserts a row at the data source for all rows in the table with a RowState [http://msdn2.microsoft.com/en-us/library/system.data.datarow.rowstate(VS.80).aspx] of Added [http://msdn2.microsoft.com/en-us/library/system.data.datarowstate (VS.80).aspx] . Inserts values for all columns that are updateable (but not columns such as identities, expressions, or timestamps).
UpdateCommand	Updates rows at the data source for all rows in the table with a **RowState** of Modified [http://msdn2.microsoft.com/en-us/library/system.data.datarowstate(VS.80).aspx] . Updates the values of all columns except for columns that are not updateable, such as identities or expressions. Updates all rows where the column values at the data source match the primary key column values of the row, and where the remaining columns at the data source match the original values of the row. For more information, see "Optimistic Concurrency Model for Updates and Deletes," later in this topic.
DeleteCommand	Deletes rows at the data source for all rows in the table with a **RowState** of Deleted [http://msdn2.microsoft.com/en-us/library/system.data.datarowstate(VS.80).aspx] . Deletes all rows where the column values match the primary key column values of the row, and where the remaining columns at the data source match the original values of the row. For more information, see "Optimistic Concurrency Model for Updates and Deletes," later in this topic.

Optimistic Concurrency Model for Updates and Deletes

The logic for generating commands automatically for UPDATE and DELETE statements is based on *optimistic concurrency*--that is, records are not locked for editing and can be modified by other users or processes at any time. Because a record could have been modified after it was returned from the SELECT statement, but before the UPDATE or DELETE statement is issued, the automatically generated UPDATE or DELETE statement contains a WHERE clause, specifying that a row is only updated if it contains all original values and has not been deleted from the data source. This is done to avoid overwriting new data. Where an automatically generated update attempts to update a row that has been deleted or that does not contain the original values found in the DataSet [http://msdn2.microsoft.com/en-us/library/system.data.dataset(VS.80).aspx] , the command does not affect any records, and a DBConcurrencyException [http://msdn2.microsoft.com/en-us/library/system.data.dbconcurrencyexception (VS.80).aspx] is thrown.

If you want the UPDATE or DELETE to complete regardless of original values, you must explicitly set the **UpdateCommand** for the **DataAdapter** and not rely on automatic command generation.

Limitations of Automatic Command Generation Logic

The following limitations apply to automatic command generation.

Unrelated Tables Only

The automatic command generation logic generates INSERT, UPDATE, or DELETE statements for stand-alone tables without taking into account any relationships to other tables at the data source. As a result, you may encounter a failure when calling **Update** to submit changes for a column that participates in a foreign key constraint in the database. To avoid this exception, do not use the **DbCommandBuilder** for updating columns involved in a foreign key constraint; instead, explicitly specify the statements used to perform the operation.

Table and Column Names

Automatic command generation logic fails if column names or table names contain any special characters, such as spaces, periods, quotation marks, or other nonalphanumeric characters, even if delimited by brackets. Fully qualified table names in the form of *catalog.schema.table* are supported.

Using the CommandBuilder to Automatically Generate an SQL Statement

To automatically generate SQL statements for a **DataAdapter**, first set the **SelectCommand** property of the **DataAdapter**, then create a **CommandBuilder** object, and specify as an argument the **DataAdapter** for which the **CommandBuilder** will automatically generate SQL statements.

Visual Basic Copy Code

```
' Assumes that connection is a valid SqlConnection object
' inside of a Using block.
Dim adapter As SqlDataAdapter = New SqlDataAdapter( _
  "SELECT * FROM dbo.Customers", connection)
Dim builder As SqlCommandBuilder = New SqlCommandBuilder(adapter)
builder.QuotePrefix = "["
builder.QuoteSuffix = "]"
```

C# Copy Code

```
// Assumes that connection is a valid SqlConnection object
// inside of a using block.
SqlDataAdapter adapter = new SqlDataAdapter(
  "SELECT * FROM dbo.Customers", connection);
SqlCommandBuilder builder = new SqlCommandBuilder(adapter);
builder.QuotePrefix = "[";
builder.QuoteSuffix = "]";
```

Modifying the SelectCommand

If you modify the **CommandText** of the **SelectCommand** after the INSERT, UPDATE, or DELETE commands have been automatically generated, an exception may occur. If the modified **SelectCommand.CommandText** contains schema information that is inconsistent with the **SelectCommand.CommandText** used when the insert, update, or delete commands were automatically generated, future calls to the **DataAdapter.Update** method may attempt to access columns that no longer exist in the current table referenced by the **SelectCommand**, and an exception will be thrown.

You can refresh the schema information used by the **CommandBuilder** to automatically generate commands by calling the **RefreshSchema** method of the **CommandBuilder**.

If you want to know what command was automatically generated, you can obtain a reference to the automatically generated commands by using the **GetInsertCommand**, **GetUpdateCommand**, and **GetDeleteCommand** methods of the **CommandBuilder** object and checking the **CommandText** property of the associated command.

The following code example writes to the console the update command that was automatically generated.

Copy Code

```
Console.WriteLine(builder.GetUpdateCommand().CommandText)
```

The following example recreates the `Customers` table in the `custDS` dataset. The **RefreshSchema** method is called to refresh the automatically generated commands with this new column information.

Visual Basic Copy Code

```
' Assumes an open SqlConnection and SqlDataAdapter inside of a Using bl
adapter.SelectCommand.CommandText = _
  "SELECT CustomerID, ContactName FROM dbo.Customers"
builder.RefreshSchema()

custDS.Tables.Remove(custDS.Tables("Customers"))
adapter.Fill(custDS, "Customers")
```

C# Copy Code

```
// Assumes an open SqlConnection and SqlDataAdapter inside of a using b
adapter.SelectCommand.CommandText =
  "SELECT CustomerID, ContactName FROM dbo.Customers";
builder.RefreshSchema();

custDS.Tables.Remove(custDS.Tables["Customers"]);
adapter.Fill(custDS, "Customers");
```

See Also

Concepts

Executing a Command [http://msdn2.microsoft.com/en-us/library/tyy0sz6b(VS.80).aspx]
Creating Commands and Retrieving Data [http://msdn2.microsoft.com/en-us/library/9hy8csk1(VS.80).aspx]

Other Resources

Working with Commands [http://msdn2.microsoft.com/en-us/library/ms254953(VS.80).aspx]